DD2424 Project
Shakespeare text generation

Wai-Hong Anton Fu, whafu@kth.se

Isac Lorentz, ilorentz@kth.se

Kunal Bhatanagar, kunalb@kth.se

George Malki, geomal@kth.se

Abstract

This project concerns the generation of Shakespeare-like text sequences using mainly LSTMs and how to evaluate these texts sequences as well as enhancing the text generation process. In this project, we found that the metrics: perplexity, term token ratio and BLEU score are most suitable for our project. We experimented with different extensions to improve the model from given metrics. One such proposed improvement was data augmentation which improved the term token ratio, however performed lesser in the perplexity score. We experimented with different sampling methods (beam search, temperature sampling, top k as well as top p) and found top k to be the best performing sampling method.

1 Introduction

Natural Language Generation (NLG) is a software that produces natural languages from data. Common applications of NLG are auto-completion, content summarization, and chatbots. Unlike classification problems, evaluating NLG is highly non-trivial, and there are many suggested methods for doing so [5, 15].

The main objective of this project is to implement LSTMs and investigate how to properly evaluate a generated text sequence from it, both *qualitatively* (surveying) and *quantitatively*. What are some strengths and weaknesses in those metrics? Furthermore, we are going to implement different text generation methods such as *top k*, *nucleus sampling*, *temperature sampling*, and *beam search*, which we will evaluate with the metrics we deem suitable. We will also investigate the effect of the overall performance when increasing or decreasing the number of hidden nodes, as well as implementing *data augmentation*.

2 Related work

The use of RNNs to randomly generate style-imitating text was made popular by a blog post made by Andrej Karpathy [4]. Furthermore, the research papers on various NLG metrics include [5, 15, 16]

3 Data

The train- and test data used in this project comes from a subset of Shakespeare plays [3]. The dataset contains about 1 155 000 words, and was split 70/30 in training- and test in a coherent way, i.e. every split consist of uninterrupted paragraphs.

In our experiment, we implemented a rule-based data augmentation method, where we did a random synonym replacement using the NLTK library [1] in Python. We augmented our Shakespeare text by appending a new Shakespeare text that was created by replacing each word with one of its synonyms in the Shakespeare text with a probability of 30 percent. The synonyms were selected randomly. After this we redid the 70/30 data split to create train and test data.

4 Method

4.1 Training the Networks

Number of Hidden Nodes	Number of Layers	Embedded Dimensions	Characters per Sequence	Training Iterations	Learning Rate
25	2	100	100	10000	0.005
50	2	100	100	10000	0.005
250	2	100	100	10000	0.005
500	2	100	100	10000	0.005

Table 1: LSTM configuration. *Character per Sequence* refers to the number of characters the LSTM is trained on, at each iteration.

We use the python neural network library *PyTorch* as our primary tool when implementing the networks. An LSTM and RNN with 100 hidden nodes each were implemented. Since it's trivial that

LSTM performs better than RNN according to various papers [8, 13], this report will only look at the smoothed loss when comparing the two networks.

Several LSTMs was implemented with different numbers of hidden nodes, the configurations can be seen in table 1 which will be used for benchmarking the effect of hidden node size, and different sentence generation methods.

Note that when building the LSTM, we also implemented a character embedding layer with 100 dimensions. Character embedding has shown to be beneficial in terms of generating out-of-vocabulary words, handling misspellings, and reducing model complexity [12]. This is however out of the scope of this report.

4.2 NLG Evaluation Metrics

NLG evaluation is usually split into two wider categories, *intrinsic*- and *extrinsic* evaluation methods [5]. Extrinsic evaluation methods test whether the machine text is successful in impacting the performance of the downstream task, which is not applicable for our case, since our NLG text doesn't have a "final goal", unlike for instance text translation. This project will instead only use intrinsic evaluation methods and metrics, which focuses on the content and quality of the machine text. Intrinsic evaluation metrics are in turn split into *qualitative*- and *quantitative* metrics.

4.2.1 Qualitative Metrics

Human evaluation is the most reliable evaluation metric [15], we will judge each text sequence by:

> **Coherence:** Is the flow of the information logical, the way connections are made (i.e. "But consider that ...") understandable, etc.
>
> **Grammaticality:** The text does not contain grammatical errors, such as "He do count to ten".
>
> **Shakespeare Similarity:** The text is similar to something that Shakespeare himself would write.

Normally in NLP tasks, other metrics such as informativeness and relevance are used. This is however not applicable for our purpose, we thus coined our own metric, *Shakespeare Similarity*. To make the evaluation unbiased, we use a survey to ask people to evaluate the generated sequences. The survey consists of 5 text sequences of 400 characters, each generated with a different method (see section 4.3). The questionnaire is asked to rate each of the five texts on the 3 qualitative metrics on a scale from 1 to 7.

One issue regarding human evaluation is that there is always a level of subjectivity, or the instructions might be ambiguous, so we'll use *Krippendorf's* α to measure the inter-evaluator agreement [5], where the α-value measures how reliable a survey result is. This was implemented using the Python library *krippendorff* to check whether our survey results are valid.

4.2.2 Quantitative Metrics

The quantitative metrics used are the following: spelling (correct spelling percentage), repetition (Term Token Ratio, aka. TTR), perplexity, and text precision (BLEU and BERTScore). By generating random sequences from our models we could calculate the correct spelling percentage and term token ratio of the generated text. By comparing the generated text sequence (the candidate text) to the test dataset (the reference text), we calculated perplexity and text precision.

Spelling Percentage To evaluate the correctness of the spelling of the generated text, a correct spelling percentage score was calculated. Using the *pyspellchecker* Python library [2], we checked our generated words against a list of correct English words and also against the compound words in the English stop words in NLTK [1] (it's, wasn't, etc).

TTR Score Some configurations of our models outputed repetitive texts. To measure the amount of repetitiveness in the text we used a simple metric called the term token ratio / lexical density which is the quotient $\frac{\textit{unique words in text}}{\textit{words in texts}}$.

Perplexity In information theory, perplexity measures how well a probability model predicts a sample. Perplexity is a measurement of the inverse probability of a generated text sequence normalized by the number of words in the sequence [6], given a known probability distribution of the words in the generated sequence. The word probability distribution comes from our test set and is a bigram probability distribution. We used a smoothed bigram word probability model for our perplexity measurement: $P(w_i|w_{i-1}) = \lambda_1 P(w_i|w_{i-1}) + \lambda_2 P(w_i) + \lambda_3$ (full equation in appendix A). The lambda values used are specified in table 2.

Table 2: Perplexity smoothing parameters

λ_1	λ_2	λ_3
$9.9e-1$	$9.999e-3$	$1e-6$

Bilingual Evaluation Understudy (BLEU) BLEU is one of the most common *n*-gram-based metrics and was first used to evaluate machine translation. It is based on the ratio of n-gram overlap between the reference and candidate text to the total number of n-grams in the candidate text. BLEU scores are typically computed for n-grams 1-4. The score can be expected to decrease as n increases. A BLEU-score over 30 generally reflect an understandable translation and a score over 50 a good and fluent translation [10]. Since BLEU has a brevity penalty (BP) to penalize candidate texts shorted than the reference, we used a modified version of the BLEU-metric, which removed this penalty, since we compare short generated sequences of text to a much longer reference text. All reported BLEU-scores in the report have this modification.

BERTScore Calculates the cosine similarity between tokens from the candidate and reference text using cosine similarity between the words in the candidate and reference text using BERT word embeddings. Using greedy matching, each word in the candidate text is paired with the most similar word in the reference text [16]. Unlike BLEU, BERTScore has no upper bound on the length of dependencies between the candidate and reference it can capture [16].

4.3 Generating sequences

For all conducted experiments, texts of length 200 characters were generated by our LSTM models. The generation was initiated by giving it a random uppercase letter. The following text generation methods were used:

Greedy search One of the easiest methods to generate a new token is to choose the token that has the highest probability for each position. This method can produce relatively acceptable results but leaves a lot to be desired when it comes to linguistic variety. We chose to omit this generation algorithm for a more reliable and relatively greedy algorithm, beam search [7].

Beam Search Beam search is a heuristic search algorithm that expands the idea of greedy search to which takes the *N* best tokens with the highest probability. Unlike Greedy search where we take consider each position in isolation, Beam search must take into account the *N* best paths that led to the current tokens. The probabilities of the combination of these paths together with the current tokens are taken into consideration to choose the most optimal sequence. The hyperparameter *N* is known as *beam width* [7].

Temperature Sampling Temperature sampling is a sampling method is inspired by statistical thermodynamics, where high temperature means low energy states are more likely encountered (low energy states equate to chars with a low probability of being sampled during a run for our model).

Equation for temperature can be found in appendix C where the value for T ranged from 0 to infinity to skew the distribution towards high probability events, which implicitly lowers the mass in the tail distribution [9]. The closer to zero we are the closer we are to sampling the most likely token. Setting T to infinity would correspond to uniform sampling. In general, high temperature would often result in greater linguistic variety but would also result in a text that is less grammatically correct.

Top *k* Sampling The main idea of top k sampling is simple, we decrease the possible sample space from which we do our sampling. We do that by first sorting the sample space by probability and then

we "remove" anything below the k'th token/highest probability. This is done by either setting their probability to $-\infty$ or to 0. The sampling is then done with respect to the probabilities of the k tokens that are left.

Top p Sampling Top p sampling or nucleus sampling, is a sampling method that takes the cumulative distribution of the expected tokens into account, instead of the top k most probable tokens which were mentioned in the previous method. This is done by first sorting the sample space by probability and then removing/cutting off the tokens where the cumulative distribution sum exceeds p. We do this by either setting their probability to $-\infty$ or to 0. The sampling is then done with respect to the probabilities of the tokens that are left. Top p sampling solves one of the main problems that plague the aforementioned top k sampling: sampling from a narrow distribution where only a few tokens make up for the majority of the acceptable tokens. By using top p sampling we avoid sampling egregiously wrong tokens and still retain some variety in our predictions.

5 Experiments

5.1 Experiment 1: Validation of Qualitative Metrics

To validate that our quantitative metrics are working correctly, we hand-picked 3 text sequences that each have their own characteristics[1], which are the following:

Text 1 Original text taken directly from the train set. Expected to score high in all metrics.

Text 2 Greedily generated from a LSTM and thus contains a lot of repetitions of words.

Text 3 Generated from a LSTM with a very high temperature, meaning it is very incoherent and composed of more or less random words.

Figure 5.1 shows that text 1 scored very well on perplexity and text 2 also scored fairly well. Text 3 scored poorly on perplexity, as expected. A similar pattern is shown on the BLEU score, although BLEU 1 and BLEU 2 seemed to score higher for text 2 in comparison to text 1. Note particularly that text 1 is the only text with a nonzero BLEU-4 score. Futhermore, text 2 scored poorly on TTR, due to its repetitive nature, which is expected. Texts 1 and 3 did well on that metric.

BERTScore did not vary significantly between the three texts. We believe that could be due to the giant size of the reference text in comparison to the candidate text. With a large reference text, it is likely that most words in the candidate text find a word in the reference text with high cosine similarly. Spelling percentage also gave sub-optimal results, likely because Shakespeare uses old English. We, therefore, believe the BertScore and spelling percentage aren't relevant metrics for our purpose.

The results indicate that perplexity might be the best metric for judging text sequences, but since it doesn't seem to take repetitive text sequences into account, it should perhaps be used in conjunction with TTR Score. BLEU scores are also useful. BertScore is likely useful in for example language translation, where we have references that directly corresponds to the generated sentence. The full result of this experiment can be found in appendix D.

Figure 1: The most relevant results from this experiment, showing the metrics *BLEU Score*, *perplexity*, and *TTR* for the 3 texts.

[1]The exact text sequences can be found in appendix B

5.2 Experiment 2: RNN vs LSTM

An RNN and LSTM were implemented, both with 2-layers and 100 hidden nodes. The results is shows that LSTM has much smaller loss than RNN (around 50% less). The exact results can be found in appendix E. Note that the LSTM and RNN comparison is not the focus of our project.

5.3 Experiment 3: Number of Hidden Nodes

This experiment consists of tuning the number of hidden nodes as our hyperparameter. Finding an optimal value for the number of hidden nodes will yield a model that has better accuracy for the given metrics used. The experiment was carried out by a grid search for the values of $m_{hidden\ nodes} = \{25, 50, 250, 500\}$. This was done by using the LSTM models described in section 4.1. The models were then evaluated on the following metrics: BertScore, BLEU score, Perplexity score, TTR Score, and smooth-loss.

> **TTR score**: Showed that the models for 500 and 25 had the highest results, indicating that they had the least repetition in of the words as well as the highest variance in the text.
>
> **Perplexity score**: There was a big difference in the perplexity score where m=25 (score>600) and 50 (score=200) had a very high perplexity score indicating the inaccuracy of the model to predict a sample of Shakespeare. However, when the hidden node size was 250 and 500 the score was 100 and approximately equal.
>
> **BLEU Score**: was approximately the same for all the models. However, m=250 did perform slightly better than the other models.
>
> **Smooth-loss**: the best models were m=250 and 500 followed by 50 and in the last position 25.

To summarize, this experiment showed that the best performing values were for hidden node sizes of 250 and 500 while 50 and 25 performed the worst. 500 hidden nodes did however receive better results than 250 nodes in regards to TTR score and perplexity. The exact result can be seen in appendix F.

5.4 Experiment 4: Temperature Sampling, Top P and Top K

The purpose of this experiment is the investigate how implementing temperature and top p sampling affects the quality of our LSTMs, which we described in table 1. Moreover, the metrics we'll use for this benchmarking are the same as shown in section 5.1, except for *BartScore*, which we deemed unfit for our purpose.

The results show that a temperature of around 0.25-0.50 is optimal in regards to perplexity and BLEU score, although not for TTR (i.e. there are a lot of repetitions of words), thus to compensate, a temperature of around 0.5-0.75 seems to be best for the overall quality of the generated text sequences.

For top p and top k sampling, the generated text sequences seem optimal when k is around 4-10 and the optimal p is around 0.4-0.7, although top p is harder to determine. The full result can be found in appendix G.

5.5 Experiment 5: Survey

A total of 27 people answered the survey (appendix I), which is, unfortunately, a quite small sample size. The mean of each metric can be seen in table 3. The results were close to what we expected when we shared the survey. We expected Beam search to be the worse but were surprised by how bad the generated text was perceived. Theoretically, the heuristic would improve upon the greedy search algorithm massively so these lacklustre results could potentially be due to a faulty implementation of the Beam search algorithm. We were also surprised that top *k* performed better than top *p* since the latter is supposed to be an improvement over the former. Though this could be explained by the fact that we chose a far too high of a p-value which could result in us sampling egregiously wrong tokens. Combining some of our sampling methods resulted in a marginal improvement over each of the sampling methods but could be affected massively by the small sample size.

For future research, it may be interesting to investigate whether there's any correlation between the qualitative- and quantitative metrics. This can likely be done using a correlations matrix of some sort. It would also be interesting to study multiple combinations of sampling methods such as (temp + top*k*) and see how they perform. To control whether our survey results are valid, table 3 show that

Table 3: The mean value of Coherence, Grammaticality, and Shakespeare Similarity, gathered from the survey (Appendix I). Moreover, the Krippendorf α for each text is shown.

	Coherence	Grammaticality	Shakespeare Similarity	Krippendorf α
Temperature = 0.25	3.458	3.666	4.750	0.350
Beam width = 3	1.500	1.500	1.875	0.499
Top *k*, k = 5	4.041	3.791	5.041	0.284
Top *p*, p = 0.95	3.166	3.291	4.291	0.176
Temp = 0.35, Top *p*, p = 0.95	3.541	3.416	4.291	0.126

most of AI-Text 4 and 5 are not reliable at all due to the low α-value. Traditionally one can draw conclusions with a data when $\alpha \geq 0.800$ and when $0.800 > \alpha \geq 0.667$ only tentative conclusions can be drawn from the data. For any $\alpha < 0.667$ then the data should be discarded [14]. However, due to the complexity of NLP, it is common practice to lower that threshold [5]. In any case, AI-Text 2 has the most agreement.

The poor α-value may suggest that qualitative metrics weren't well defined or that the instructions were unclear. However, evaluating NLP typically involves trained evaluators that are experts in their fields, rather than crowdsourcing [5]. In conclusion, crowdsourcing may not be of good practice for qualitative evaluation of NLP.

5.6 Experiment 6: Data Augmentation

In order to see the effects that data augmentation can have on the performance, we trained an LSTM model with 200 nodes in the hidden layer with 2-hidden layers on the augmented training dataset and compared it to a model with the same hyperparameters trained on the non-augmented Shakespeare text. The models were trained for 10000 iterations. We evaluated each model on smooth loss, perplexity, and TTR.

The results from smooth loss indicated that the model trained on the augmented data performed worse than the model trained on the non-augmented data. Hence, the score indicated that the model trained on augmented data performed worse, appendix H. The output result of 400 characters can be seen as text 1 (out from model trained on augmented data) and text 2 (output from model trained on non-augmented data) in appendix H.

The same results could be seen in the perplexity. The model trained on the augmented data received a score of 199.5 and the model trained on the non-augmented data received a score of 189.0. This score was averaged over 100 test runs for the perplexity. To note, the bigram data for the perplexity was created from the non-augmented Shakespeare test set. Hence we can see that the result indicated that the model trained on the non-augmented dataset performed better on predicting a sample of Shakespeare. The reason for a lower score of the model trained on the augmented dataset might be that the data-augmentations were not confined to the domain of Shakespeare texts. When doing data-augmentation we do modification to the data under the assumption that these modification are valid for the domain, such as rotating an image in a model that classifies objects since the object should still be recognized. However, in the field of Shakespeare replacing a word enclosed in the Shakespeare domain with another word that might be outside of the domain will result in a worsened score. A better approach might have been using Neural augmentation to get domain specific words [11].

The results for the TTR showed that the model trained on the augmented-dataset received a score of 0.59 and the model trained on the non-augmented dataset received 0.57. This behavior could be explained due to the fact that we have added words in the augmented dataset that were not present in the original Shakespeare text. Resulting in a increase in variety of words produced.

6 Conclusion

The conclusions are included in each experiment.

References

[1] Natural language toolkit. https://www.nltk.org/. Accessed: 2022-05-18.

[2] pyspellchecker. https://github.com/barrust/pyspellchecker. Accessed: 2022-05-18.

[3] Shakespeare dataset. https://storage.googleapis.com/download.tensorflow.org/data/shakespeare.txt. Accessed: 2022-05-18.

[4] Andrej. The unreasonable effectiveness of recurrent neural networks. http://karpathy.github.io/2015/05/21/rnn-effectiveness/. Accessed: 2022-05-18.

[5] Asli Celikyilmaz, Elizabeth Clark, and Jianfeng Gao. Evaluation of text generation: A survey, 2020. URL https://arxiv.org/abs/2006.14799.

[6] J H Martin D Jurafsky. *Speech and Language Processing*. 3rd ed. draft edition, 2022.

[7] Ketan Doshi. Foundations of nlp explained visually: Beam search, how it works, May 2021. URL https://towardsdatascience.com/foundations-of-nlp-explained-visually-beam-search-how-it-works-1586b9849a24.

[8] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9: 1735–80, 12 1997. doi: 10.1162/neco.1997.9.8.1735.

[9] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration, 2019. URL https://arxiv.org/abs/1904.09751.

[10] Alon Lavie. Evaluating the output of machine translation systems. https://www.cs.cmu.edu/~alavie/Presentations/MT-Evaluation-MT-Summit-Tutorial-19Sep11.pdf. Accessed: 2022-05-18.

[11] Kenton Lee, Kelvin Guu, Luheng He, Tim Dozat, and Hyung Won Chung. Neural data augmentation via example extrapolation, 2021. URL https://arxiv.org/abs/2102.01335.

[12] Edward Ma. Besides word embedding, why you need to know character embedding?, Jun 2018. URL https://towardsdatascience.com/besides-word-embedding-why-you-need-to-know-character-embedding-6096a34a3b10.

[13] Ralf C. Staudemeyer and Eric Rothstein Morris. Understanding lstm – a tutorial into long short-term memory recurrent neural networks, 2019. URL https://arxiv.org/abs/1909.09586.

[14] Wikipedia. Krippendorff's alpha — Wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=Krippendorff's%20alpha&oldid=1022617836, 2022. [Online; accessed 22-May-2022].

[15] Weizhe Yuan, Graham Neubig, and Pengfei Liu. Bartscore: Evaluating generated text as text generation, 2021.

[16] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert, 2020.

A Appendix: Smoothed Bigram Perplexity Formula

For a generated sequence $W = w_1 w_2 ... w_N$, a smoothed bigram perplexity is defined as:

$$PP(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i|w_{i-1})}}, P(w_i|w_{i-1}) = \lambda_1 P(w_i|w_{i-1}) + \lambda_2 P(w_i) + \lambda_3 \quad (1)$$

B Appendix: Experiment 1 text

Text 1

```
MENENIUS:
Sir, I shall tell you. With a kind of smile,
Which ne’er came from the lungs, but even thus--
For, look you, I may make the belly smile
As well as speak--it tauntingly replied
To the discontented members, the mutinous parts
That envied his receipt; even so most fitly
As you malign our senators for that
They are not such as you.
```

Text 2

```
Second Servant:
The lords to the father to the provost of the word to the souls of the souls of the so to the liv
And that the souls to the world to the provost to the provost to the world to the some to the sou
```

Text 3

```
Fiseraou:
My now! depost there head give voult’s bacfontly
To good, a greefordicorte;--
You neo, live--
Serancuher naice you gone goal in Frother,
Hethy brother breaty a tropphoss, aloneus pot’s wiffoods:
ever by.
```

C Appendix: Temprature equation

$$p(j) = \frac{exp(x_j/T)}{\sum_{i=1}^{k}(x_i/T)} \tag{2}$$

D Appendix: Experiment 1 Results

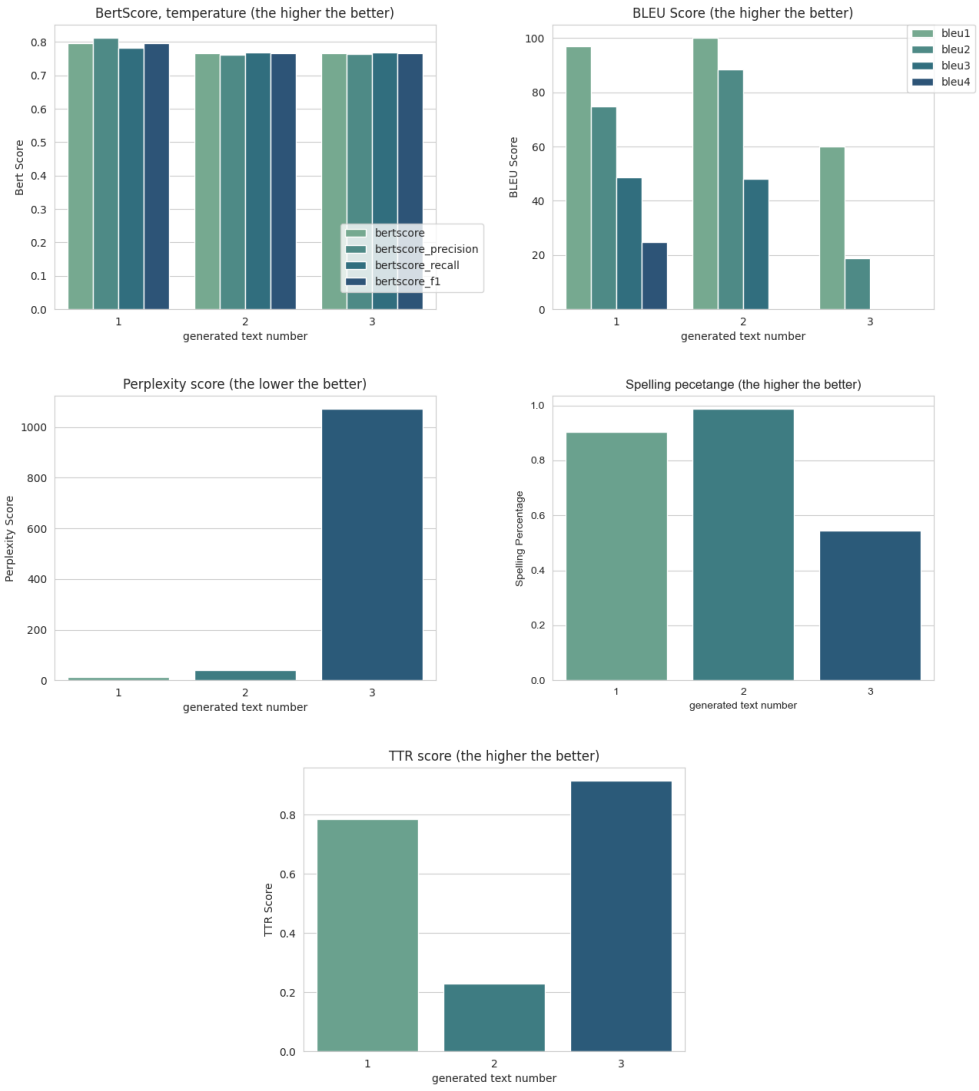

Figure 2: Figures showing how the 3 texts scored on the metrics *BartScore*, *BLEU Score*, *Perplexity*, *Spelling Percentage* and *TTR* for the 3 texts.

E Appendix: Experiment 2 Results

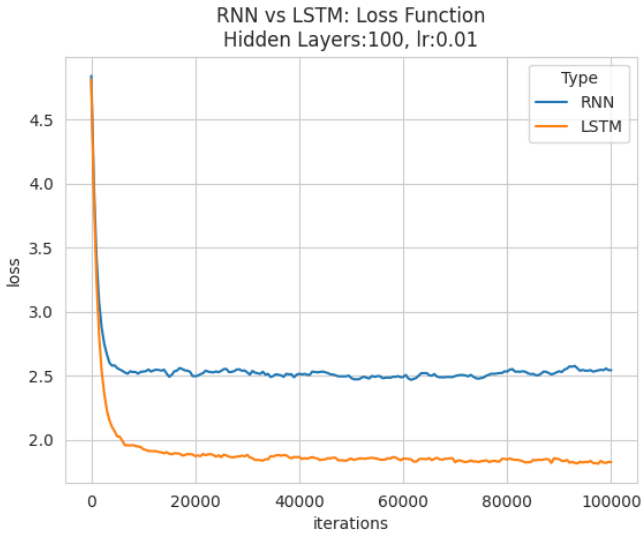

Figure 3: Plot of loss functions for RNN and LSTM.

F Appendix: Experiment 3 Results

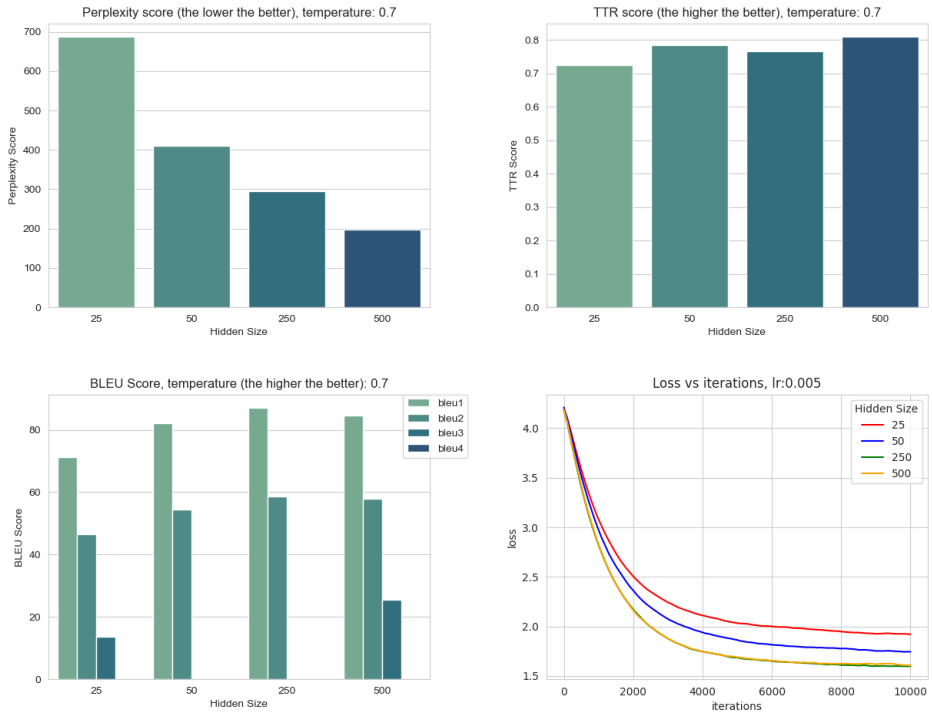

Figure 4: Showing the scores of the text generated from the different LSTMs in regards to, *perplexity*, *TTR score*, the 4 *BLEU Scores*, as well as their smoothed loss over iterations

G Appendix: Experiment 4 Results

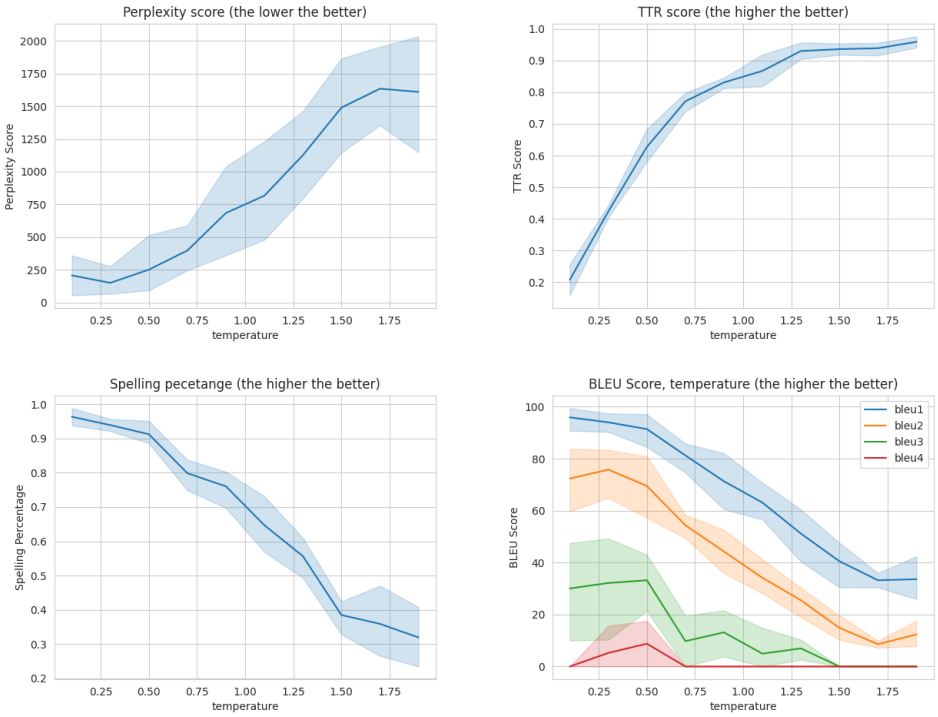

Figure 5: Figure showing how LSTM performs in regards of the 4 most relevant quantitative metrics, *perplexity*, *TTR score*, *Spelling Percentage* and the 4 *BLEU Scores* when changing the **temperature**. Each line the mean of the 4 hidden layer size (25, 50, 250 and 500 hidden nodes) while the coloured area around the line represents its confidence interval.

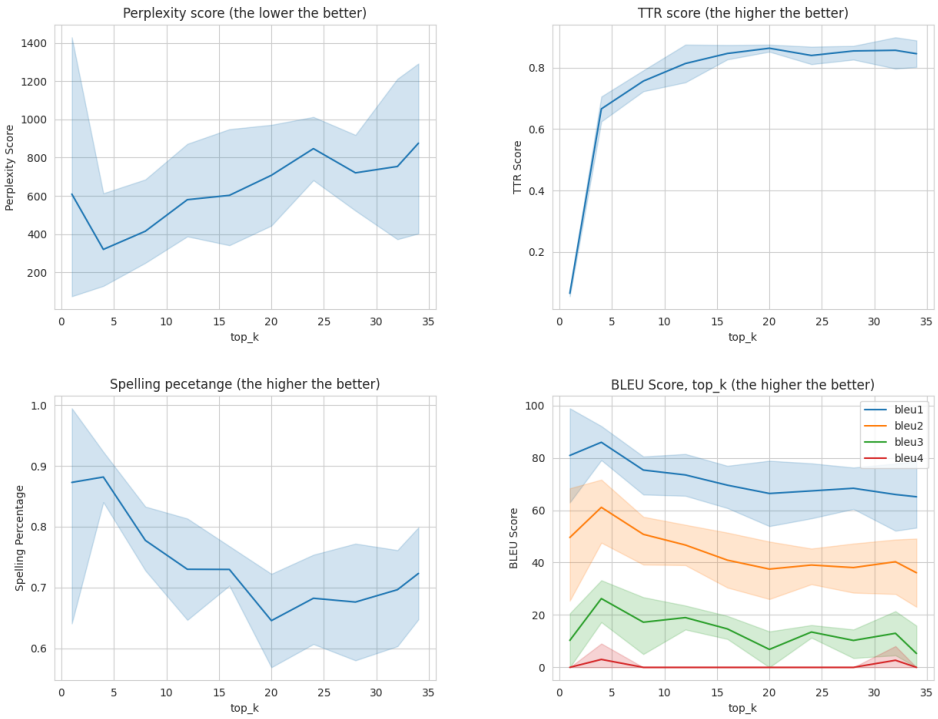

Figure 6: Figure showing how LSTM performs in regards of the 4 most relevant quantitative metrics, *perplexity*, *TTR score*, *Spelling Percentage* and the 4 *BLEU Scores* when changing the **top K (Nucleus Sampling)**. Each line the mean of the 4 hidden layer size (25, 50, 250 and 500 hidden nodes) while the coloured area around the line represents its confidence interval.

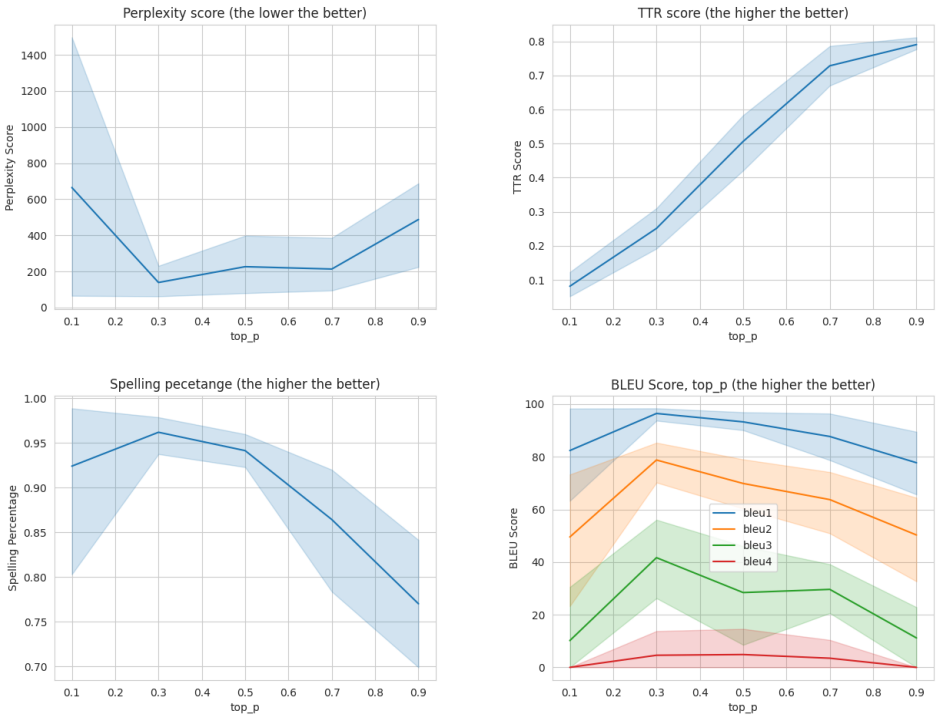

Figure 7: Figure showing how LSTM performs in regards of the 4 most relevant quantitative metrics, *perplexity*, *TTR score*, *Spelling Percentage* and the 4 *BLEU Scores* when changing the **top P**. Each line the mean of the 4 hidden layer size (25, 50, 250 and 500 hidden nodes) while the coloured area around the line represents its confidence interval.

H Appendix: Experiment 6

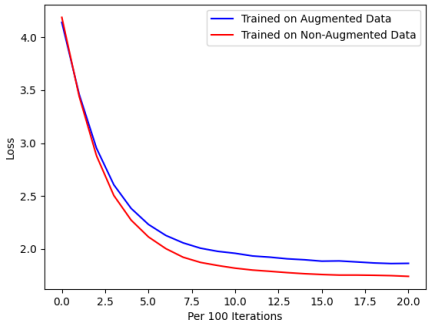

Figure 8: Showing the scores of the smooth loss for model trained on augmented data and model trained on non-augmented data.

Text 1

```
the from and souson to the not of him the from there
I shall me dead in woo prove the a the carrom the most
whom the more I will being the good to the come to me.

First the good for the glore
The word how did me and power to be the love to my more.

First I am the sence.

First Servose this the come the come hold be mought,
And prespeace.

First Call the good
But that with the grace it what desp
```

Text 2

```
thou should the tore the very, and a charge the desere have the world,
And all the some of the mack of the wear of the should so both the would
I call the more to should be no good for the
should as be rouson the was the write of the say:
And show I will the disour's for parth and the lose how and the greath
Such me the horse of the souse and the word, I countences him,
Thou waster and the trust he
```

I Appendix: Qualitative Evaluation Survey

Quality evaluation of Shakespeare text generated from AI (Artificial Intelligence)

This survey is a part of our project in the KTH course DD2424 (Deep Learning in Data science), where we randomly generate text sequences from several different AI models that have been trained on Shakespeare scripts. You will be asked to read 4 short texts and rate them according to 3 metrics, Coherence, Grammaticality and Shakespeare Similarity. The definitions of these metrics are:

Coherence:
Is the flow of the information logical, the way connections are made (i.e. "But consider that ...") understandable etc.

Grammaticality:
The text does not contain grammatical errors, such as "He do count to ten".

Shakespeare Similarity:
The text is similar to something that Shakespeare himself would write. For reference, below is a passage taken from "Romeo and Juliet" by Shakespeare:

ROMEO:
Ha, banishment! be merciful, say 'death;'
For exile hath more terror in his look,
Much more than death: do not say 'banishment.'

FRIAR LAURENCE:
Hence from Verona art thou banished:
Be patient, for the world is broad and wide.

Thanks for your participation!

* Required

AI Text 1

My lords that he doth to the sorrow to have not shall so some to the fair to he to the death,
And thou would to the sorrow to the souls with the distress to me to the will,
And that the life of the father somether so have somether to my rater to the father.

LUCIO:
This like to the words to my son.

1. Coherence (AI Text 1) *

Is the flow of the information logical, the way connections are made (i.e. But consider that ...) understandable etc.

Mark only one oval.

	1	2	3	4	5	6	7	
Not coherent at all	◯	◯	◯	◯	◯	◯	◯	Very coherent

2. Grammaticality (AI Text 1) *

The text does not contain grammatical errors, such as "He do count to ten".

Mark only one oval.

	1	2	3	4	5	6	7	
Not grammatically correct at all	◯	◯	◯	◯	◯	◯	◯	Very gramatically correct

Reference text to Shakespeare Similarity. Below is a text taken from "Romeo and Juliet" by Shakespeare.

ROMEO:
Ha, banishment! be merciful, say 'death;'
For exile hath more terror in his look,
Much more than death: do not say 'banishment.'

FRIAR LAURENCE:
Hence from Verona art thou banished:
Be patient, for the world is broad and wide.

3. Shakespeare Similarity (AI Text 1) *

The text is similar to something that Shakespeare himself would write. The reference can be found above.

Mark only one oval.

	1	2	3	4	5	6	7	
Not similar at all	◯	◯	◯	◯	◯	◯	◯	Very similar

AI Text 2

Matoth ad i te apot. til's,
DELOLA: adiu,
DERY BLWAD OFLUE: oriu,
LOTES EDA: orad,
LATETRETUFETADE: adiu,
DERYCENETETETES Sor is ary avil o tu trath ad tatis
HAS: ito tuly, menadedede ad tatis
HAS: ito tuly, menadedede ad tatis
HAS: ito tuly, menadedede ad tatis
HAS: ito tuly, menadedede ad tatis

4. Coherence (AI Text 2) *

Is the flow of the information logical, the way connections are made (i.e. But consider that ...) understandable etc.

Mark only one oval.

	1	2	3	4	5	6	7	
Not coherent at all	◯	◯	◯	◯	◯	◯	◯	Very coherent

5. Grammaticality (AI Text 2) *

The text does not contain grammatical errors, such as "He do count to ten".

Mark only one oval.

	1	2	3	4	5	6	7	
Not grammatically correct at all	◯	◯	◯	◯	◯	◯	◯	Very grammatically correct

Reference text to Shakespeare Similarity. Below is a text taken from "Romeo and Juliet" by Shakespeare.

ROMEO:
Ha, banishment! be merciful, say 'death;'
For exile hath more terror in his look,
Much more than death: do not say 'banishment.'

FRIAR LAURENCE:
Hence from Verona art thou banished:
Be patient, for the world is broad and wide.

6. Shakespeare Similarity (AI Text 2) *

The text is similar to something that Shakespeare himself would write. The reference can be found above.

Mark only one oval.

	1	2	3	4	5	6	7	
Not similar at all	◯	◯	◯	◯	◯	◯	◯	Very similar

AI Text 3

My lord, a more and stoning, would thoughts: stand that to my deed;
And my days, into me in saw to my daughter, we may, soult myself.

Secord Satirent:
Will that he willoor, iffititus on that heaster and true his wife of thou sade an eart
As stand is a shall were and with thee to thy leaves we desire

7. Coherence (AI Text 3) *

Is the flow of the information logical, the way connections are made (i.e. But consider that ...) understandable etc.

Mark only one oval.

	1	2	3	4	5	6	7	
Not coherent at all	◯	◯	◯	◯	◯	◯	◯	Very coherent

8. Grammaticality (AI Text 3) *

The text does not contain grammatical errors, such as "He do count to ten".

Mark only one oval.

	1	2	3	4	5	6	7	
Not grammatically correct at all	◯	◯	◯	◯	◯	◯	◯	Very grammatically correct

Reference text to Shakespeare Similarity. Below is a text taken from "Romeo and Juliet" by Shakespeare.

ROMEO:
Ha, banishment! be merciful, say 'death;'
For exile hath more terror in his look,
Much more than death: do not say 'banishment.'

FRIAR LAURENCE:
Hence from Verona art thou banished:
Be patient, for the world is broad and wide.

9. Shakespeare Similarity (AI Text 3) *

The text is similar to something that Shakespeare himself would write. The reference can be found above.

Mark only one oval.

	1	2	3	4	5	6	7	
Not similar at all	◯	◯	◯	◯	◯	◯	◯	Very similar

AI Text 4

My seem in God, inforing much
Done a prown dripe is to wive sword?

Secome, thou metter's too:
And have three with cappors?

BUCKINGHAM:
A my fing have to alone more
town time with him to wrong
As a loa with with safes you thanks;
Lords. Now, regate in lip in dying as restrand and see thy heat:
Can w

10. Coherence (AI Text 4) *

Is the flow of the information logical, the way connections are made (i.e. But consider that ...) understandable etc.

Mark only one oval.

	1	2	3	4	5	6	7	
Not coherent at all	◯	◯	◯	◯	◯	◯	◯	Very coherent

11. Grammaticality (AI Text 4) *

The text does not contain grammatical errors, such as "He do count to ten".

Mark only one oval.

	1	2	3	4	5	6	7	
Not grammatically correct at all	◯	◯	◯	◯	◯	◯	◯	Very grammatically correct

Reference text to Shakespeare Similarity. Below is a text taken from "Romeo and Juliet" by Shakespeare.

ROMEO:
Ha, banishment! be merciful, say 'death;'
For exile hath more terror in his look,
Much more than death: do not say 'banishment.'

FRIAR LAURENCE:
Hence from Verona art thou banished:
Be patient, for the world is broad and wide.

12. Shakespeare Similarity (AI Text 4) *

The text is similar to something that Shakespeare himself would write. The reference can be found above.

Mark only one oval.

	1	2	3	4	5	6	7	
Not similar at all	◯	◯	◯	◯	◯	◯	◯	Very similar

AI Text 5

My some to the good breath in the like to the provost of the world
And to the say that a will he to the provost to the live with with the world of the words,
So so so too this with her souls of the hath to the provost of him to the stand the love.

GLOUCESTER:
I have him some that we may a daughters

13. Coherence (AI Text 5) *

Is the flow of the information logical, the way connections are made (i.e. But consider that ...) understandable etc.

Mark only one oval.

	1	2	3	4	5	6	7	
Not coherent at all	◯	◯	◯	◯	◯	◯	◯	Very coherent

14. Grammaticality (AI Text 5) *

The text does not contain grammatical errors, such as "He do count to ten".

Mark only one oval.

	1	2	3	4	5	6	7	
Not grammatically correct at all	◯	◯	◯	◯	◯	◯	◯	Very grammatically correct

Reference text to Shakespeare Similarity. Below is a text taken from "Romeo and Juliet" by Shakespeare.

ROMEO:
Ha, banishment! be merciful, say 'death;'
For exile hath more terror in his look,
Much more than death: do not say 'banishment.'

FRIAR LAURENCE:
Hence from Verona art thou banished:
Be patient, for the world is broad and wide.

15. Shakespeare Similarity (AI Text 5) *

The text is similar to something that Shakespeare himself would write. The reference can be found above.

Mark only one oval.

	1	2	3	4	5	6	7	
Not similar at all	◯	◯	◯	◯	◯	◯	◯	Very similar